Robotics: Science and Systems 2020
Corvalis, Oregon, USA, July 12-16, 2020

Quantile QT-Opt for Risk-Aware Vision-Based Robotic Grasping

Cristian Bodnar*, Adrian Li†, Karol Hausman‡, Peter Pastor† and Mrinal Kalakrishnan†
*Department of Computer Science and Technology
University of Cambridge, Cambridge, UK
Work was done while an AI Resident at X
Email: cb2015@cam.ac.uk
†X, Mountain View, California, USA
‡Google Brain, Mountain View, California, USA

***Abstract*—The distributional perspective on reinforcement learning (RL) has given rise to a series of successful Q-learning algorithms, resulting in state-of-the-art performance in arcade game environments. However, it has not yet been analyzed how these findings from a discrete setting translate to complex practical applications characterized by noisy, high dimensional and continuous state-action spaces. In this work, we propose Quantile QT-Opt (Q2-Opt), a distributional variant of the recently introduced distributed Q-learning algorithm [12] for continuous domains, and examine its behaviour in a series of simulated and real vision-based robotic grasping tasks. The absence of an actor in Q2-Opt allows us to directly draw a parallel to the previous discrete experiments in the literature without the additional complexities induced by an actor-critic architecture. We demonstrate that Q2-Opt achieves a superior vision-based object grasping success rate, while also being more sample efficient. The distributional formulation also allows us to experiment with various risk distortion metrics that give us an indication of how robots can concretely manage risk in practice using a Deep RL control policy. As an additional contribution, we perform batch RL experiments in our virtual environment and compare them with the latest findings from discrete settings. Surprisingly, we find that the previous batch RL findings from the literature obtained on arcade game environments do not generalise to our setup.**

I. Introduction

The new distributional perspective on RL has produced a novel class of Deep Q-learning methods that learn a distribution over the state-action returns, instead of using the expectation given by the traditional value function. These methods, which obtained state-of-the-art results in the arcade game environments [4, 6, 5], present several attractive properties.

First, their ability to preserve the multi-modality of the action values naturally accounts for learning from a non-stationary policy, most often deployed in a highly stochastic environment. This ultimately results in a more stable training process and improved performance and sample efficiency. Second, they enable the use of risk-sensitive policies that no longer select actions based on the expected value, but take entire distributions into account. These policies can represent a continuum of risk management strategies ranging from risk-averse to risk-seeking by optimizing for a broader class of risk metrics.

Despite the improvements distributional Q-learning algorithms demonstrated in the discrete arcade environments, it is yet to be examined how these findings translate to practical, real-world applications. Intuitively, the advantageous properties of distributional Q-learning approaches should be particularly beneficial in a robotic setting. The value distributions can have a significant qualitative impact in robotic tasks, usually characterized by highly-stochastic and continuous state-action spaces. Additionally, performing safe control in the face of uncertainty is one of the biggest impediments to deploying robots in the real world, an impediment that RL methods have not yet tackled. In contrast, a distributional approach can allow robots to learn an RL policy that appropriately quantifies risks for the task of interest.

However, given the brittle nature of deep RL algorithms and their often counter-intuitive behaviour [9], it is not entirely clear if these intuitions would hold in practice. Therefore, we believe that an empirical analysis of distributional Q-learning algorithms in real robotic applications would shed light on their benefits and scalability, and provide essential insight for the robot learning community.

In this paper, we aim to address this need and perform a thorough analysis of distributional Q-learning algorithms in simulated and real vision-based robotic manipulation tasks. To this end, we propose a distributional enhancement of QT-Opt [12] subbed Quantile QT-Opt (Q2-Opt). The choice of QT-Opt, a recently introduced distributed Q-learning algorithm that operates on continuous action spaces, is dictated by its demonstrated applicability to large-scale vision-based robotic experiments. In addition, by being an actor-free generalization of Q-learning in continuous action spaces, QT-Opt enables a direct comparison to the previous results on the arcade environments without the additional complexities and compounding effects of an actor-critic-type architecture.

In particular, we introduce two versions of Q2-Opt, based on Quantile Regression DQN (QR-DQN) [6] and Implicit Quantile Networks (IQN) [5]. The two methods are evaluated on a vision-based grasping task in simulation and the real world. We show that these distributional algorithms achieve state-of-the-art grasping success rate in both settings, while also being more sample efficient. Furthermore, we experiment

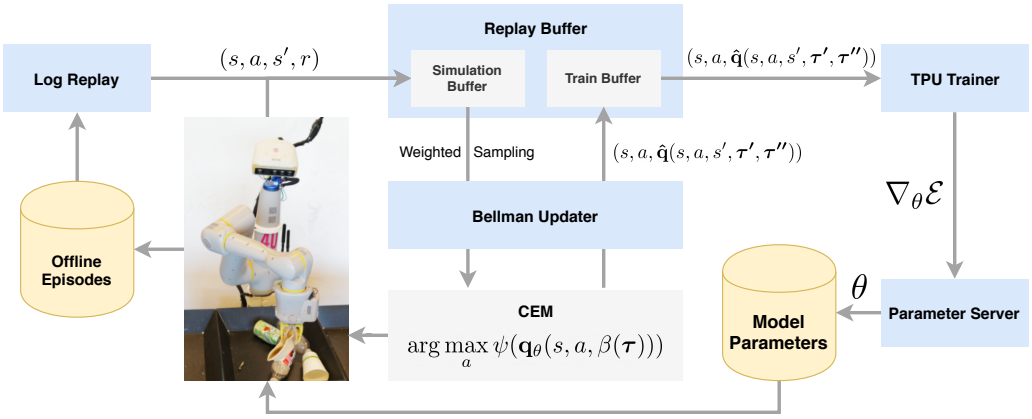


Fig. 1: Distributed system architecture of Q2-Opt. The interactions between the robot and the environment are either stored in a database of episodes for later offline learning, or they are directly sent to the replay buffer when online learning is performed. The samples from the Simulation Buffer are pulled by the Bellman Updater, which appends the distributional targets. These labelled transitions are pushed to the train buffer and consumed by the TPU Training workers to compute the gradients. The parameter server uses the gradients to update the weights, which are asynchronously pulled by the agents.

with a multitude of risk metrics, ranging from risk-seeking to risk-averse, and show that risk-averse policies can bring significant performance improvements. We also report on the interesting qualitative changes that different risk metrics induce in the robots' grasping behaviour. As an additional contribution, we analyze our distributional methods in a batch RL scenario and compare our findings with an equivalent experiment from the arcade environments [1].

II. Related Work

Deep learning has shown to be a useful tool for learning visuomotor policies that operate directly on raw images. Examples include various manipulation tasks, where related approaches use either supervised learning to predict the probability of a successful grasp [18, 15, 20] or learn a reinforcement-learning control policy [16, 22, 12].

Distributional Q-learning algorithms have been so far a separate line of research, mainly evaluated on game environments. These algorithms replace the expected return of an action with a distribution over the returns and mainly vary by the way they parametrize this distribution. Bellemare et al. [4] express it as a categorical distribution over a fixed set of equidistant points. Their algorithm, C51, minimizes the KL-divergence to the projected distributional Bellman target. A follow-up algorithm, QR-DQN [6], approximates the distribution by learning the outputs of the quantile function at a fixed set of points, the quantile midpoints. This latter approach has been extended by IQN [5], which reparametrized the critic network to take as input any probability τ and learn the quantile function itself. Besides this extension, their paper also analyses various risk-sensitive policies that the distributional formulation enables. In this work, we apply these advancements to a challenging vision-based real-world grasping task with a continuous action-space.

Closest to our work is D4PG [3], a distributed and distributional version of DDPG [17] that achieves superior performance to the non-distributional version in a series of simulated continuous-control environments. In contrast to this work, we analyze a different variant of Q-learning with continuous action spaces, which allows us to focus on actor-free settings that are similar to the previous distributional Q-learning algorithms. Besides, we demonstrate our results on real robots on a challenging vision-based grasping task.

III. Background

As previously stated, we build our method on top of QT-Opt [12], a distributed Q-learning algorithm suitable for continuous action spaces. QT-Opt is one of the few scalable deep RL algorithms with demonstrated generalisation performance in a challenging real-world task. As the original paper shows, it achieves an impressive 96% vision-based grasp success rate on unseen objects. Therefore, building on top of QT-Opt is a natural choice for evaluating value distributions on a challenging real-world task, beyond simulated environments. Additionally, by directly generalising Q-Learning to continuous domains, QT-Opt allows us to compare our results with the existing distributional literature on discrete environments.

In this paper, we consider a standard Markov Decision Process [21] formulation $(\mathcal{S}, \mathcal{A}, r, p, \gamma)$, where $s \in \mathcal{S}$ and $a \in \mathcal{A}$ denote the state and action spaces, $r(s, a)$ is a deterministic reward function, $p(\cdot|s, a)$ is the transition function and $\gamma \in (0, 1)$ is the discount factor. QT-Opt trains a parameterized state-action value function $Q_\theta(s, a)$ which is represented by a neural network with parameters θ. The cross-entropy method (CEM) [24] is used to iteratively optimize and select the best action for a given Q-function:

$$\pi_\theta(s) = \arg\max_a Q_\theta(s, a) \tag{1}$$

In order to train the Q-function, a separate process called the "Bellman Updater" samples transition tuples (s, a, r, s') containing the state s, action a, reward r, and next state s' from

a replay buffer and generates Bellman target values according to a clipped Double Q-learning rule [8, 25]:

$$\hat{Q}(s,a,s') = r(s,a) + \gamma V(s') \tag{2}$$

where $V(s') = Q_{\bar{\theta}_1}(s', \pi_{\bar{\theta}_2}(s'))$, and $\bar{\theta}_1$ and $\bar{\theta}_2$ are the parameters of two delayed target networks. These target values are pushed to another replay buffer $\mathcal{D}$, and a separate training process optimizes the Q-function against a training objective:

$$\mathcal{E}(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[D(Q_\theta(s,a), \hat{Q}(s,a,s'))\right] \tag{3}$$

where D is a divergence metric.

In particular, the cross-entropy loss is chosen for D, and the output of the network is passed through a sigmoid activation to ensure that the predicted Q-values are inside the unit interval.

IV. Quantile QT-Opt (Q2-Opt)

In Q2-Opt (Figure 1) the value function no longer predicts a scalar value, but rather a vector $\boldsymbol{q}_\theta(s,a,\boldsymbol{\tau})$ that predicts the quantile function output for a vector of input probabilities $\boldsymbol{\tau}$, with $\boldsymbol{\tau}_i \in [0,1]$ and $i = 1,\ldots,N$. Thus the i-th element of $\boldsymbol{q}_\theta(s,a,\boldsymbol{\tau})$ approximates $F^{-1}_{s,a}(\boldsymbol{\tau}_i)$, where $F^{-1}_{s,a}$ is the inverse CDF of the random action-value associated with the state-action pair (s,a). However, unlike QT-Opt where CEM optimizes directly over the Q-values, in Quantile QT-Opt, CEM maximizes a scoring function $\psi : \mathbb{R}^N \to \mathbb{R}$ that maps the vector $\boldsymbol{q}$ to a score $\psi(\boldsymbol{q})$:

$$\pi_\theta(s,\boldsymbol{\tau}) = \arg\max_a \psi(\boldsymbol{q}_\theta(s,a,\boldsymbol{\tau})) \tag{4}$$

Similarly, the target values produced by the "Bellman Updater" are vectorized using a generalization of the clipped Double Q-learning rule from QT-Opt:

$$\begin{aligned} \hat{\boldsymbol{q}}_{\bar{\theta}}(s,a,s',\boldsymbol{\tau}',\boldsymbol{\tau}'') &= r(s,a)\mathbf{1} + \gamma \boldsymbol{v}(s',\boldsymbol{\tau}',\boldsymbol{\tau}'') \\ \boldsymbol{v}(s',\boldsymbol{\tau}',\boldsymbol{\tau}'') &= \boldsymbol{q}_{\bar{\theta}_1}(s', \pi_{\bar{\theta}_2}(s',\boldsymbol{\tau}''),\boldsymbol{\tau}') \end{aligned} \tag{5}$$

where $\mathbf{1}$ is a vector of ones, and, as before, $\bar{\theta}_1$ and $\bar{\theta}_2$ are the parameters of two delayed target networks. Even though this update rule has not been considered so far in the distributional RL literature, we find it effective in reducing the overestimation in the predictions.

In the following sections, we present two versions of Q2-Opt based on two recently introduced distributional algorithms: QR-DQN and IQN. The main differences between them arise from the inputs $\boldsymbol{\tau}, \boldsymbol{\tau}'$, and $\boldsymbol{\tau}''$ that are used. To avoid overloading our notation, from now on we omit the parameter subscript in $\boldsymbol{q}_\theta, \hat{\boldsymbol{q}}_{\hat{\theta}}$ and replace it with an index into these vectors $\boldsymbol{q}_i, \hat{\boldsymbol{q}}_j$.

A. Quantile Regression QT-Opt (Q2R-Opt)

In Quantile Regression QT-Opt (Q2R-Opt), the vectors $\boldsymbol{\tau}, \boldsymbol{\tau}', \boldsymbol{\tau}''$ in $\boldsymbol{q}$ and $\hat{\boldsymbol{q}}$ are fixed. They all contain N quantile midpoints of the value distribution. Concretely, $\boldsymbol{q}_i(s,a,\boldsymbol{\tau})$ is assigned the fixed quantile target $\boldsymbol{\tau}_i = \frac{\bar{\tau}_{i-1}+\bar{\tau}_i}{2}$ with $\bar{\tau}_i = \frac{i}{N}$. The scoring function $\psi(\cdot)$ takes the mean of this vector, reducing the N quantile midpoints to the expected value of the distribution. Because $\boldsymbol{\tau}, \boldsymbol{\tau}', \boldsymbol{\tau}''$ are always fixed we consider them implicit and omit adding them as an argument to $\boldsymbol{q}$ and $\hat{\boldsymbol{q}}$ for Q2R-Opt.

The quantile heads are optimized by minimizing the Huber [10] quantile regression loss:

$$\begin{aligned} \rho^\kappa_\tau(\delta_{ij}) &= |\tau - \mathbb{I}\{\delta_{ij} < 0\}|\mathcal{L}_\kappa(\delta_{ij}) \\ \mathcal{L}_\kappa(\delta_{ij}) &= \begin{cases} \frac{1}{2}\delta^2_{ij}, & \text{if } |\delta_{ij}| \le \kappa \\ \kappa(|\delta_{ij}| - \frac{1}{2}\kappa), & \text{otherwise} \end{cases} \end{aligned} \tag{6}$$

for all the pairwise TD-errors:

$$\delta_{ij} = \hat{\boldsymbol{q}}_j(s,a,s') - \boldsymbol{q}_i(s,a) \tag{7}$$

Thus, the network is trained to minimize the loss function:

$$\mathcal{E}(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\sum_{i=1}^{N} \mathbb{E}_j[\rho^\kappa_{\boldsymbol{\tau}_i}(\delta_{ij})]\right] \tag{8}$$

We set κ, the threshold between the quadratic and linear regime of the loss, to 0.002 across all of our experiments.

B. Quantile Function QT-Opt (Q2F-Opt)

In Q2F-Opt, the neural network itself approximates the quantile function of the value distribution, and therefore it can predict the inverse CDF for any $\boldsymbol{\tau}$. Since $\boldsymbol{\tau}, \boldsymbol{\tau}', \boldsymbol{\tau}''$ are no longer fixed, we explicitly include them in the arguments of $\boldsymbol{q}$ and $\hat{\boldsymbol{q}}$. Thus, the TD-errors δ_{ij} take the form:

$$\delta_{ij} = \hat{\boldsymbol{q}}_j(s,a,s',\boldsymbol{\tau}',\boldsymbol{\tau}'') - \boldsymbol{q}_i(s,a,\boldsymbol{\tau}), \tag{9}$$

where $\boldsymbol{\tau}_i \sim U[0,1]$, $\boldsymbol{\tau}'_j \sim U[0,1]$ and $\boldsymbol{\tau}''_j \sim U[0,1]$ are sampled from independent uniform distributions. Using different input probability vectors also decreases the correlation between the networks. Note that now the length of the prediction and target vectors are determined by the lengths of $\boldsymbol{\tau}$ and $\boldsymbol{\tau}'$. The model is optimized using the same loss function as the one from Equation 8.

C. Risk-Sensitive Policies

The additional information provided by a value distribution compared to the (scalar) expected return gives birth to a broader class of policies that go beyond optimizing for the expected value of the actions. Concretely, the expectation can be replaced with any risk metric, that is any function that maps the random return to a scalar quantifying the risk. In Q2-Opt, this role is played by the function ψ that acts as a risk-metric. Thus the agent can handle the intrinsic uncertainty of the task in different ways depending on the specific form of ψ. It is important to specify that this uncertainty is generated by the environment dynamics $p(\cdot|s,a)$ and the (non-stationary) policy collecting the real robot rollouts, and that it is not a parametric uncertainty.

We distinguish two methods to construct risk-sensitive policies for Q2R-Opt and Q2F-Opt, each specific to one of the methods. In Q2R-Opt, risk-averse and risk-seeking policies can be obtained by changing the function $\psi(\cdot)$ when selecting actions. Rather than computing the mean of the target quantiles, $\psi(\cdot)$ can be defined as a weighted average over the quantiles $\psi(\boldsymbol{q}(s,a)) = \frac{1}{N}\sum w_i \boldsymbol{q}_i(s,a)$. This sum produces a

policy that is in between a worst-case and best-case action selector and, for most purposes, it would be preferable in practice over the two extremes. For instance, a robot that would consider only the worst-case scenario would most likely terminate immediately since this strategy, even though it is not useful, does not incur any penalty. Behaviours like this have been encountered in our evaluation of very conservative policies.

In contrast, Q2F-Opt provides a more elegant way of learning risk-sensitive control policies by using risk distortion metrics [26]. Recently, Majmidar and Pavone [19] have argued for the use of risk distortion metrics in robotics. They proposed a set of six axioms that any risk measure should meet to produce reasonable behaviour and showed that risk distortion metrics satisfy all of them. However, to the best of our knowledge, they have not been tried on real robotic applications.

The key idea is to use a policy:

$$\pi_\theta(s, \beta(\boldsymbol{\tau})) = \arg\max_a \psi(\boldsymbol{q}(s, a, \beta(\boldsymbol{\tau}))),$$

where $\beta : [0,1] \to [0,1]$ is an element-wise function that distorts the uniform distribution that $\boldsymbol{\tau}$ is effectively sampled from, and $\psi(\cdot)$ computes the mean of the vector as usual. Functions β that are concave induce risk-averse policies, while convex function induce risk-seeking policies.

Risk Metric	Formula
CPW	$\tau^\eta / (\tau^\eta + (1-\tau)^\eta)^{\frac{1}{\eta}}$
Wang	$\Phi(\Phi^{-1}(\tau) + \eta)$
CVaR	$\eta\tau$
Norm	$\frac{1}{\eta}\sum_i^\eta \tau_i, \tau_i \sim U[0,1]$
Pow	$\mathbb{I}_{\eta\geq 0}\tau^{\frac{1}{1+\lvert\eta\rvert}} + \mathbb{I}_{\eta<0}\left[1-(1-\tau)^{\frac{1}{1+\lvert\eta\rvert}}\right]$

TABLE I: The considered risk distortion metrics $\beta(\tau;\eta)$ with input τ and parametrised by η. Φ denotes the CDF of the standard normal distribution, and $\mathbb{I}$ is an indicator function.

In our experiments, we consider the same risk distortion metrics used by Dabney et al. [5]: the cumulative probability weighting (**CPW**) [7], the standard normal CDF-based metric proposed by **Wang** [27], the conditional value at risk (**CVaR**) [23], **Norm** [5], and a power law formula (**Pow**) [5]. Concretely, we use these metrics with a parameter choice similar to that of Dabney et al. [5]. CPW(0.71) is known to be a good model for human behaviour [28]; Wang(-0.75), Pow(-2), CVaR(0.25) and CVaR(0.4) are risk-averse. Norm(3) decreases the weight of the distribution's tails by averaging 3 uniformly sampled $\boldsymbol{\tau}$. Ultimately, Wang(0.75) produces risk-seeking behaviour. We include all these metrics in Table I.

Due to the relationships between Q2F-Opt and the literature of risk distortion measures, we focus our risk-sensitivity experiments on the metrics mentioned above and leave the possibility of trying different functions $\psi(\cdot)$ in Q2R-Opt for future work.

D. Model Architecture

To maintain our comparisons with QT-Opt, we use very similar architectures for Q2R-Opt and Q2F-Opt. For Q2R-Opt, we modify the output layer of the standard QT-Opt architecture to be a vector of size $N = 100$, rather than a scalar. For Q2F-Opt, we take a similar approach to Dabney et al. [5], and embed every $\boldsymbol{\tau}_k$ with $k \in \{1, \ldots, N = 32\}$ using a series of $n = 64$ cosine basis functions:

$$\phi_j(\boldsymbol{\tau}_k) := \text{ReLU}\left(\sum_{i=0}^{n-1} \cos(\pi i \boldsymbol{\tau}_k) w_{ij} + b_j\right)$$

We then perform the Hadamard product between this embedding and the convolutional features. Another difference in Q2F-Opt is that we replace batch normalization [11] in the final fully-connected layers with layer normalization [2]. We notice that this better keeps the sampled values in the range allowed by our MDP formulation. The three architectures are all included in a single diagram in Figure 2.

V. Results

In this section, we present our results on simulated and real environments. In simulation, we perform both online and offline experiments, while for the real world, the training is exclusively offline. We begin by describing our evaluation method.

A. Experimental Setup

We consider the problem of vision-based robotic grasping for our evaluations. In our grasping setup, the robot arm is placed at a fixed distance from a bin containing a variety of objects and tasked with grasping any object. The MDP specifying our robotic manipulation task provides a simple binary reward to the agent at the end of the episode: 0 for a failed grasp, and 1 for a successful grasp. To encourage the robot to grasp objects as fast as possible, we use a time step penalty of -0.01 and a discount factor $\gamma = 0.9$. The state is represented by a 472×472 RGB image; the actions are a mixture of continuous 4-DOF tool displacements in x, y, z with azimuthal rotation ϕ, and discrete actions to open and close the gripper, as well as to terminate the episode.

In simulation, we train the agent to grasp from a bin containing 8 to 12 randomly generated procedural objects (Figure 2). For the first $5,000$ global training steps, we use a procedural exploration policy. The scripted policy is lowering the end effector at a random position at the level of the bin and attempts to grasp. After $5,000$ steps, we switch to an ϵ-greedy policy with $\epsilon = 0.2$. We train the network from scratch (no pretraining) using Adam [13] with a learning rate 10^{-4} and batch size 4096 (256 per chip on a 4×4 TPU). Additionally, we use two iterations of CEM with 64 samples for each.

In the real world, we train our model offline from a 72 TiB dataset of real-world experiences collected over five months, containing $559,642$ episodes of up to 20 time steps each. Out of these, 39% were generated by noise-free trained QT-Opt policies, 22% by an ϵ-greedy strategy using trained QT-Opt policies and 39% by an ϵ-greedy strategy based on a scripted

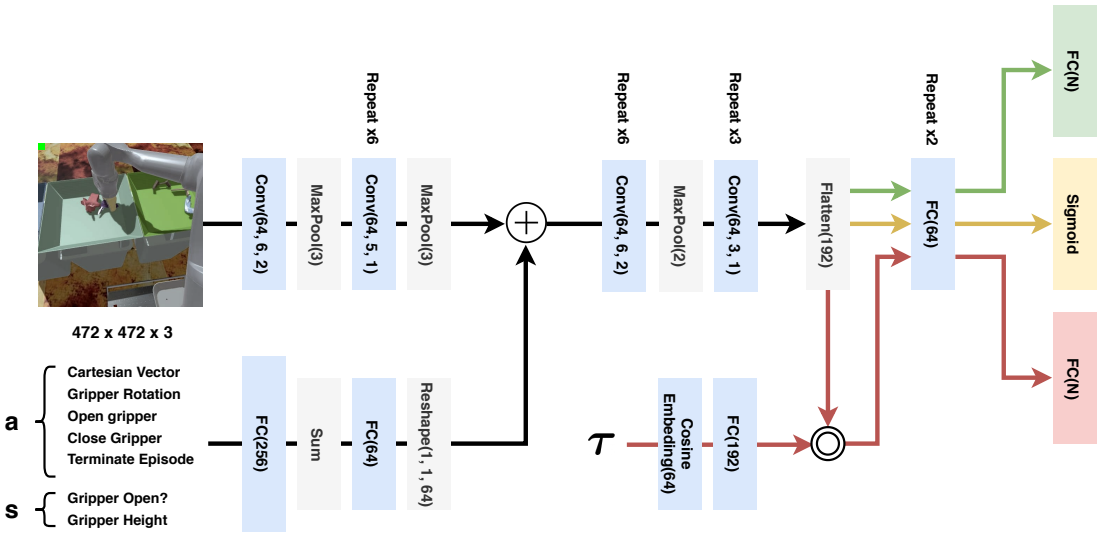


Fig. 2: The neural network architectures for QT-Opt (yellow), Q2R-Opt (green) and Q2F-Opt (red). The common components of all the three models are represented by black arrows. The top-left image shows a view from the robot camera inside our simulation environment.

policy. For evaluation, we attempt 6 consecutive grasps from a bin containing 6 objects without replacement, repeated across 5 rounds. Figure 1 includes our workspace setup. We perform this experiment in parallel on 7 robots, resulting in a total of 210 grasp attempts. All the robots use a similar object setup consisting of two plastic bottles, one metal can, one paper bowl, one paper cup, and one paper cup sleeve. In the results section, we report the success rate over the 210 attempts. Videos for the real-world experiments can be found on the website[1] accompanying the paper.

Our evaluation methodology is different from that of Kalashnikov et al. [12]. The original QT-Opt paper reports an average success rate of 76% for grasping 28 objects over 30 attempts without replacement, trained on a mixture of off-policy and on-policy data. While not directly comparable, we reproduce QT-Opt on a completely different robot with different objects and report an average success rate of 70% for grasping 6 objects over 6 attempts without replacement, trained on off-policy data only.

B. Simulation Experiments

We begin by evaluating Q2-Opt against QT-Opt in simulation. Figure 3 shows the mean success rate as a function of the global training step together with the standard deviation across five runs for QT-Opt, Q2R-Opt and Q2F-Opt. Because Q2-Opt and QT-Opt are distributed systems, the global training step does not directly match the number of environment episodes used by the models during training. Therefore, to understand the sample efficiency of the algorithm, we also include in Figure 4 the success rate as a function of the total number of environment episodes added to the buffer.

[1]https://q2-opt.github.io/

The distributional methods achieve higher success rates while also being more sample efficient than QT-Opt. While Q2F-Opt performs best, Q2R-Opt exhibits an intermediary performance and, despite being less sample efficient than Q2F-Opt, it still learns significantly faster than our baseline.

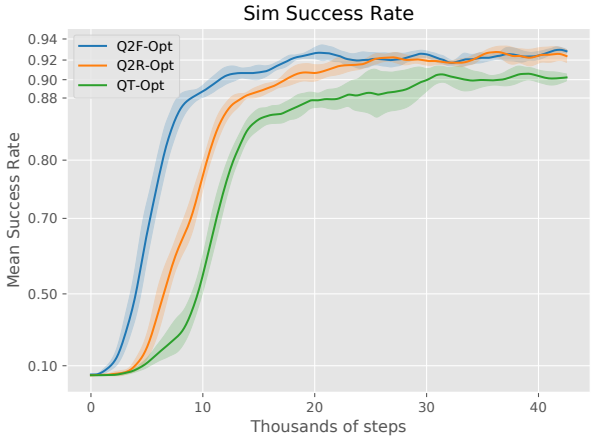


Fig. 3: Sim success rate as a function of the global step. The distributional methods achieve higher grasp success rates in a lower number of global steps.

We extend these simulation experiments with a series of risk distortion measures equipped with different parameters. Figure 5 shows the success rate for various measures used in Q2F-Opt. We notice that risk-averse policies (Wang(-0.75), Pow(-2), CVaR) are generally more stable in the late stages of training and achieve a higher success rate. Pow(-2) remarkably achieves 95% grasp success rate. However, being too conservative can also be problematic. Particularly, the CVaR(0.25) policy becomes more vulnerable to the locally

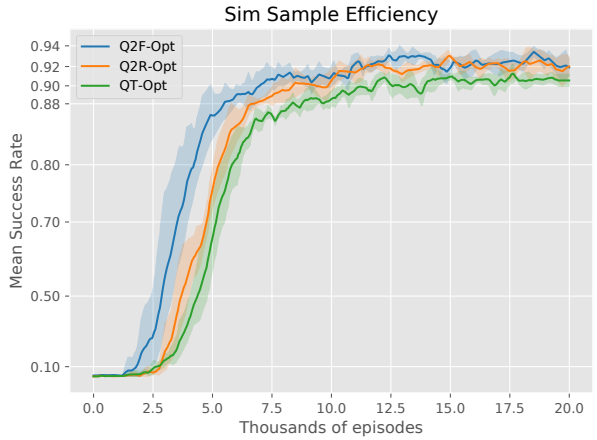


Fig. 4: Sim success rate as a function of the number of generated environment episodes. The distributional methods are significantly more sample efficient than QT-Opt.

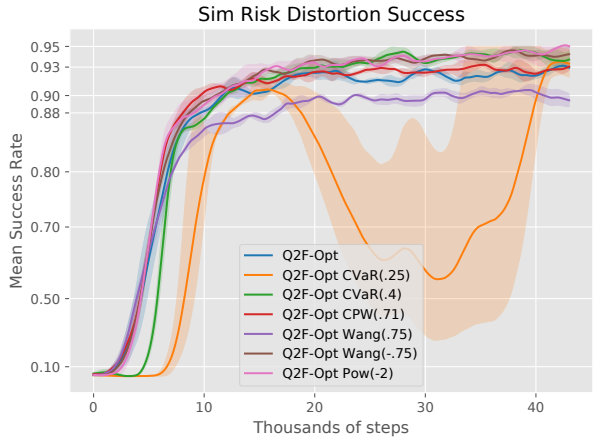


Fig. 5: Average success rate over five runs for different risk-sensitive policies in sim. Most risk-averse policies perform better, but extremely conservative ones like CVaR(0.25) can become unstable. The risk-seeking policy Wang(0.75) performs worst.

optimal behaviour of stopping immediately (which does not induce any reward penalty). This makes its performance fluctuate throughout training, even though it ultimately obtains a good final success rate. Table II gives the complete final success rate statistics.

C. Real-World Experiments

The chaotic physical interactions specific to real-world environments and the diversity of policies used to gather the experiences make the real environment an ideal setting for distributional RL. Furthermore, this experiment is also of practical importance for robotics since any increase in grasp success rate from offline data reduces the amount of costly online training that has to be performed to obtain a good policy.

We report in Table III the grasp success rate statistics

Model	Success	Std	Median
QT-Opt	0.903	0.005	0.903
Q2R-Opt	0.923	0.006	0.924
Q2F-Opt	**0.928**	**0.001**	**0.928**
Q2F-Opt Wang(0.75)	0.898	0.012	0.893
Q2F-Opt CPW(0.71)	0.928	0.003	0.925
Q2F-Opt CVAR(0.25)	0.933	0.013	0.941
Q2F-Opt CVAR(0.4)	0.938	0.008	0.938
Q2F-Opt Wang(-0.75)	0.942	0.007	0.944
Q2F-Opt Pow(-2.0)	**0.950**	**0.004**	**0.952**

TABLE II: Final sim success rate statistics. Distributional risk-averse policies have the best performance.

Model	Grasp Success Rate
QT-Opt	70.00%
Q2R-Opt	79.50%
Q2F-Opt	**82.00%**
Q2F-Opt CPW(0.71)	75.71%
Q2F-Opt Wang(0.75)	78.10%
Q2F-Opt Norm(3)	80.47%
Q2F-Opt Pow(-2)	83.81%
Q2F-Opt CVaR(0.4)	85.23%
Q2F-Opt Wang(-0.75)	**87.60%**

TABLE III: Real world grasp success rate out of 210 total grasps. Our methods significantly outperform QT-Opt, while risk-averse polices are better by a significant margin.

for all the considered models. We find that the best risk-averse version of Q2-Opt achieves an impressive 17.6% higher success rate than QT-Opt. While the real evaluation closely matches the model hierarchy observed in sim, the success rate differences between the models are much more significant.

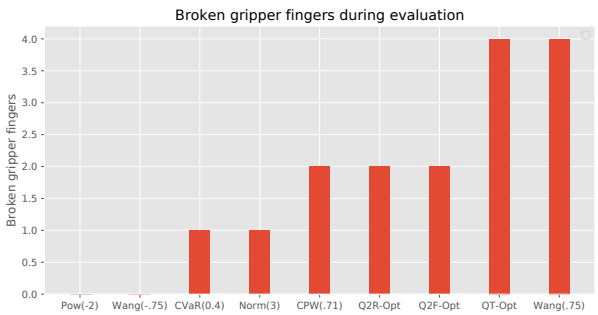
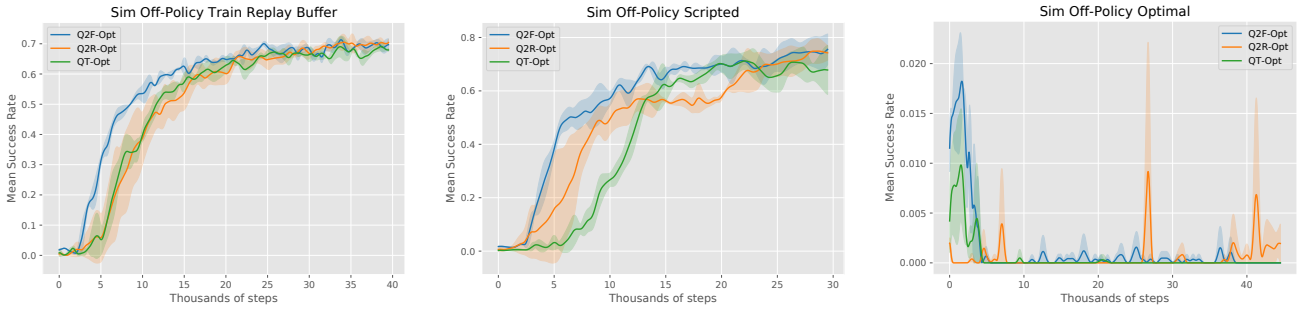


Fig. 6: The number of detached gripper fingers during evaluation. Among the risk-averse policies (first three), only one gripper broke. The agents controlled by risk-neutral policies (Q2R-Opt, Q2F-Opt, QT-Opt) lost eight gripper fingers in total, with half of those belonging to QT-Opt. The last policy, which is risk-seeking lost four, similar to QT-Opt. The other policies (Norm, CPW) behaved similarly to risk-neutral policies.

Besides the improvement in performance, we notice that the distortion measures of Q2F-Opt have a significant qualitative impact, even though the training is performed from the same data. Risk-averse policies tend to readjust the gripper in positions that are more favourable or move objects around to



(a) Sim off-policy success rate for data collected during a full training run. None of the methods can achieve the final performance of the policy trained online (90%).

(b) Sim success rate from an offline dataset produced by a scripted exploration policy. The models achieve a higher success rate than on the replay buffer dataset.

(c) Sim success rate from an offline dataset produced by a policy close to optimality. None of the models is able to learn from a dataset of successful grasps.

Fig. 7: Batch RL experiments in simulation. Distributional methods show little to no improvement in our simulated environment.

make grasping easier. CVaR(0.4), the most conservative metric we tested in the real world, presented a particularly interesting behaviour of intentionally dropping poorly grasped objects to attempt a better re-grasp. The CVaR policy mainly used this technique when attempting to grasp objects from the corners of the bin to move them in a central position.

However, a downside of risk-averse policies that we noticed is that, for the difficult-to-grasp paper cup sleeves, the agent often kept searching for an ideal position without actually attempting to grasp. We believe this is an interesting example of the trade-offs between being conservative and risk-seeking. The only tested risk-seeking policy, using Wang(0.75), made many high-force contacts with the bin and objects, which often resulted in broken gripper fingers and objects being thrown out of the bin. We also showcase some of these qualitative differences between the considered policies in the videos accompanying our paper.

These qualitative differences in behaviour that value distributions cause can provide a way to achieve safe robot control. An interesting metric we considered to quantify these behaviours is the number of broken gripper fingers throughout the entire evaluation process presented above. Occasionally, the gripper fingers break in high-force contacts with the objects or the bin. Figure 6 plots these numbers for each policy. Even though we do not have a statistically significant number of samples, we believe this figure is a good indicator that risk-averse policies implicitly achieve safer control.

D. Batch RL and Exploitation

Recently, Agarwal et al. [1] have argued that most of the advantages of distributional algorithms come from better exploitation. Their results demonstrated that QR-DQN could achieve in offline training a performance superior to online C51. Since environment interactions are particularly costly in robotics, we aim to reproduce these results in a robotic setting. Therefore, we perform an equivalent experiment in simulation and train the considered models on all the transitions collected during training by a QT-Opt agent with a final success rate of 90% (Figure 7a).

We note that despite the minor success rate improvements brought by Q2R-Opt and Q2F-Opt, the two models are not even capable of achieving the final success rate of the policy trained from the same data. We hypothesize this is due to the out-of-distribution action problem [14], which becomes more prevalent in continuous action spaces.

We investigated this further on two other datasets: one collected by a scripted stochastic exploration policy with 46% success rate and another produced by an almost optimal policy with 89% grasp success rate. Figures 7b and 7c plot the results for these two datasets. Surprisingly, the models achieve a higher success rate on the scripted exploration dataset than on the dataset collected during training. On the dataset generated by the almost optimal policy, none of the methods manages to obtain a reasonable success rate. These results, taken together with our real-world experiments, suggest that offline datasets must contain a diverse set of experiences to learn effectively in a batch RL setting.

VI. Conclusion

In this work, we have examined the impact that value distributions have on practical robotic tasks. Our proposed methods, collectively called Q2-Opt, achieved state-of-the-art success rates on simulated and real vision-based robotic grasping tasks, while also being significantly more sample efficient than the non-distributional equivalent, QT-Opt. Additionally, we have shown how safe reinforcement learning control can be achieved through risk-sensitive policies and reported the rich set of behaviours these policies produce in practice despite being trained from the same data. As a final contribution, we evaluated the proposed distributional methods in a batch RL setting similar to that of Agarwal et al. [1] and showed that, unfortunately, their findings do not translate to the continuous grasping environment presented in this work.

Acknowledgments

We would like to give special thanks to Ivonne Fajardo and Noah Brown for overseeing the robot operations. We would also like to extend our gratitude to Julian Ibarz for helpful comments.

References

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. Striving for simplicity in off-policy deep reinforcement learning. *arXiv preprint arXiv:1907.04543*, 2019.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

[4] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458, 2017.

[5] Will Dabney, Georg Ostrovski, David Silver, and Remi Munos. Implicit quantile networks for distributional reinforcement learning. In *International Conference on Machine Learning*, pages 1104–1113, 2018.

[6] Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[7] Rodrigo González and George Wu. On the shape of the probability weighting function. *Cognitive Psychology*, 38:129–166, 1999.

[8] Hado V. Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

[9] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[10] Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[12] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673, 2018.

[13] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[14] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11761–11771, 2019.

[15] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[16] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[17] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[18] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. 2017.

[19] Anirudha Majumdar and Marco Pavone. How should a robot assess risk? Towards an axiomatic theory of risk in robotics. In *Robotics Research: The 18th International Symposium ISRR*, page 75. Springer Nature, 2017.

[20] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3406–3413. IEEE, 2016.

[21] Martin L. Puterman. Markov decision processes: Discrete stochastic dynamic programming. In *Wiley Series in Probability and Statistics*, 1994.

[22] Deirdre Quillen, Eric Jang, Ofir Nachum, Chelsea Finn, Julian Ibarz, and Sergey Levine. Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6284–6291. IEEE, 2018.

[23] R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–41, 2000.

[24] Reuven Y. Rubinstein and Dirk P. Kroese. The cross-entropy method. In *Information Science and Statistics*, 2004.

[25] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

[26] Shaun Wang. Premium calculation by transforming the layer premium density. *ASTIN Bulletin: The Journal of the IAA*, 26(1):71–92, 1996.

[27] Shaun S. Wang. A class of distortion operators for pricing financial and insurance risks. *Journal of Risk and Insurance*, pages 15–36, 2000.

[28] George Wu and Richard Gonzalez. Curvature of the probability weighting function. *Management Science*, 42 (12):1676–1690, 1996.